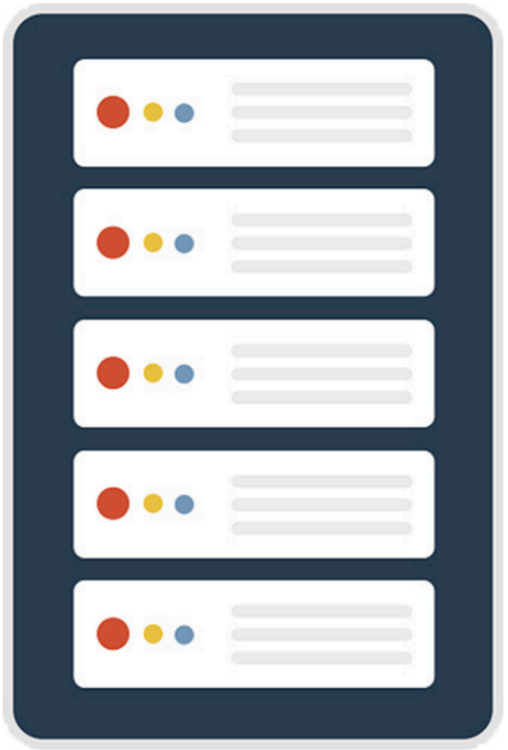
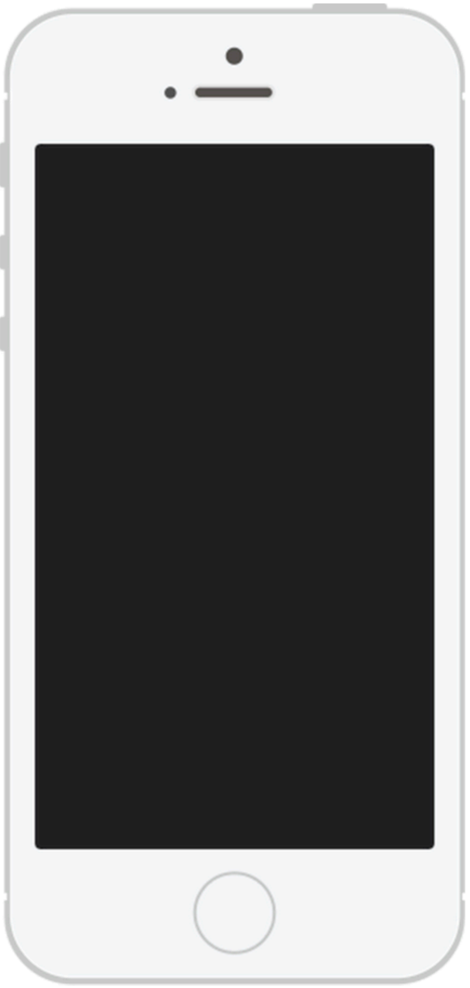
Why GraphQL?

StarWars API

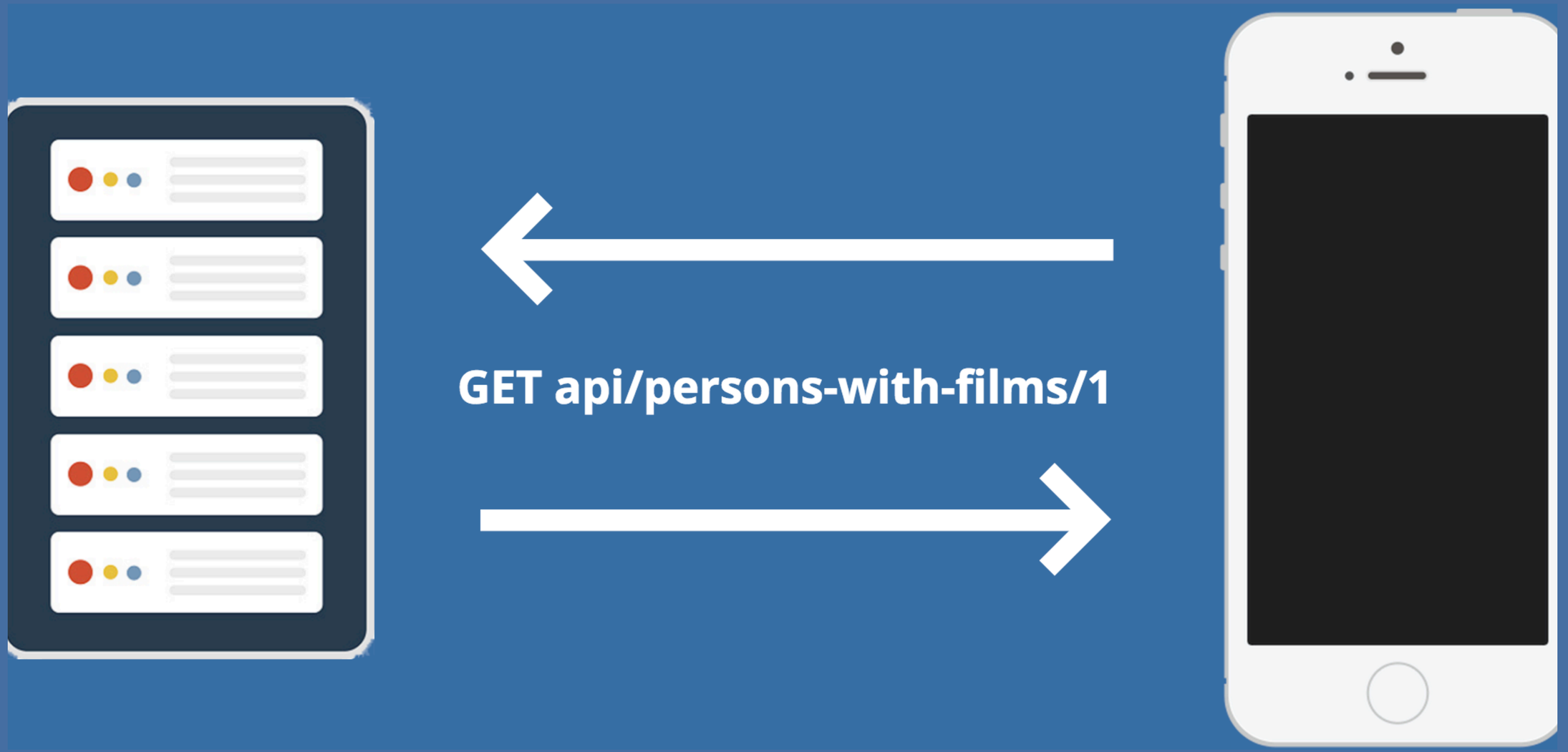


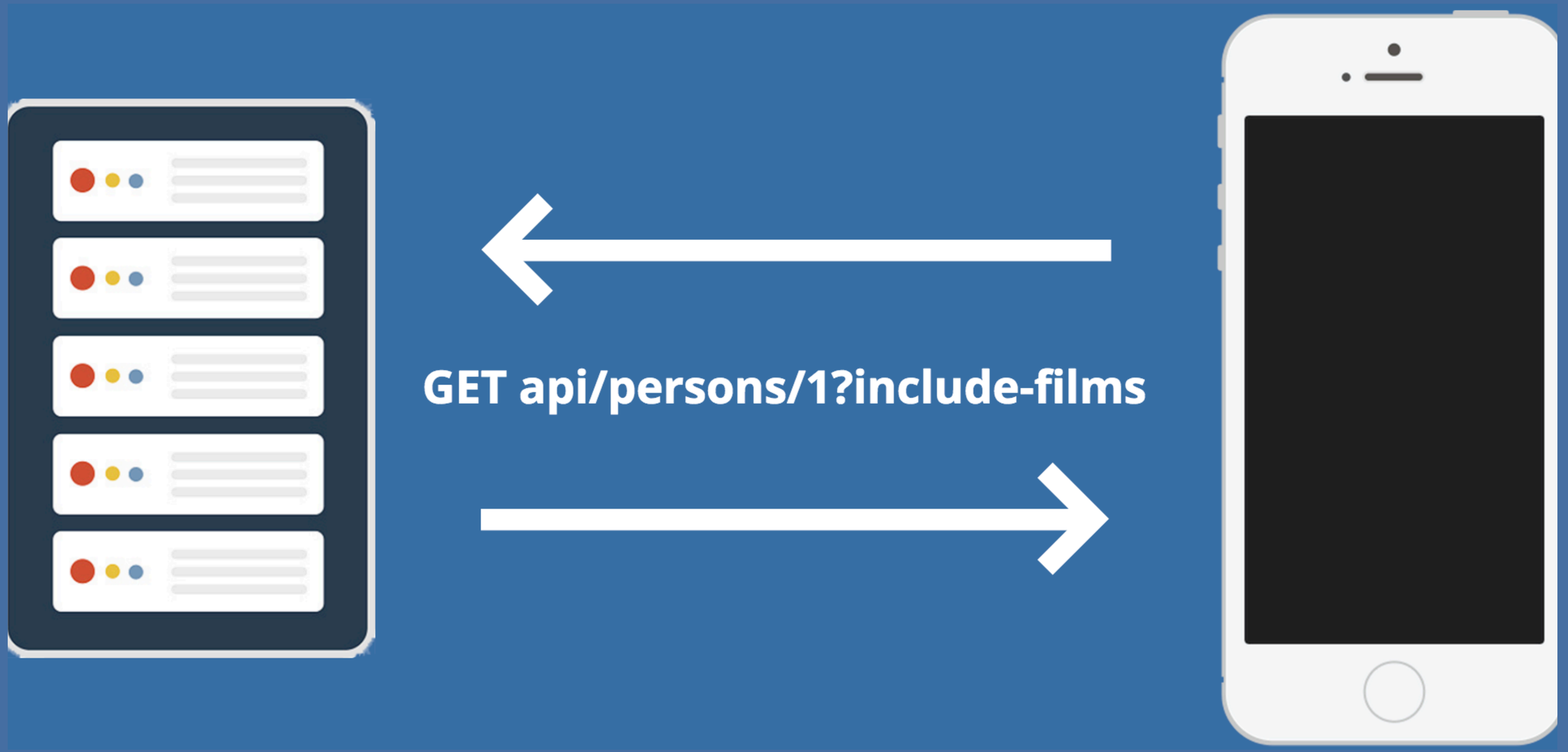
GET api/people/1



```
{
  "name": "Luke Skywalker",
  "height": "1.72 m",
  "mass": "77 Kg",
  "hair_color": "Blond",
  "skin_color": "Caucasian",
  "eye_color": "Blue",
  "birth_year": "19 BBY",
  "gender": "Male",
  "homeworld": "http://swapi.co/api/planets/1/",
  "films": [
    "http://swapi.co/api/films/1/",
    "http://swapi.co/api/films/2/",
    "http://swapi.co/api/films/3/"
  ],
  "species": ["http://swapi.co/api/species/1/"],
  "vehicles": [
    "http://swapi.co/api/vehicles/14/",
    "http://swapi.co/api/vehicles/30/"
  ],
  "starships": [
    "http://swapi.co/api/starships/12/",
    "http://swapi.co/api/starships/22/"
  ],
  "created": "2014-12-09T13:50:51.644000Z",
  "edited": "2014-12-10T13:52:43.172000Z",
  "url": "http://swapi.co/api/people/1/"
}
```

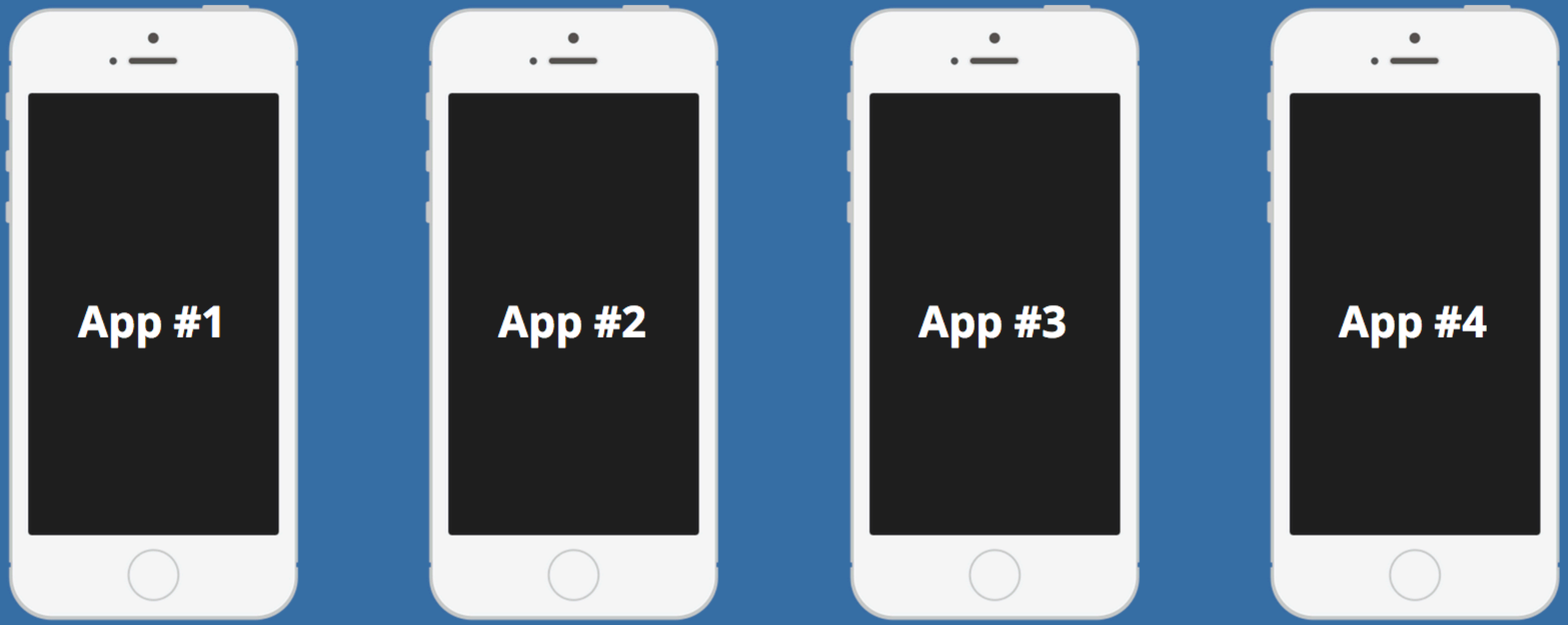
GET api/films/1
GET api/films/2
GET api/films/3


GET api/persons-with-films/1

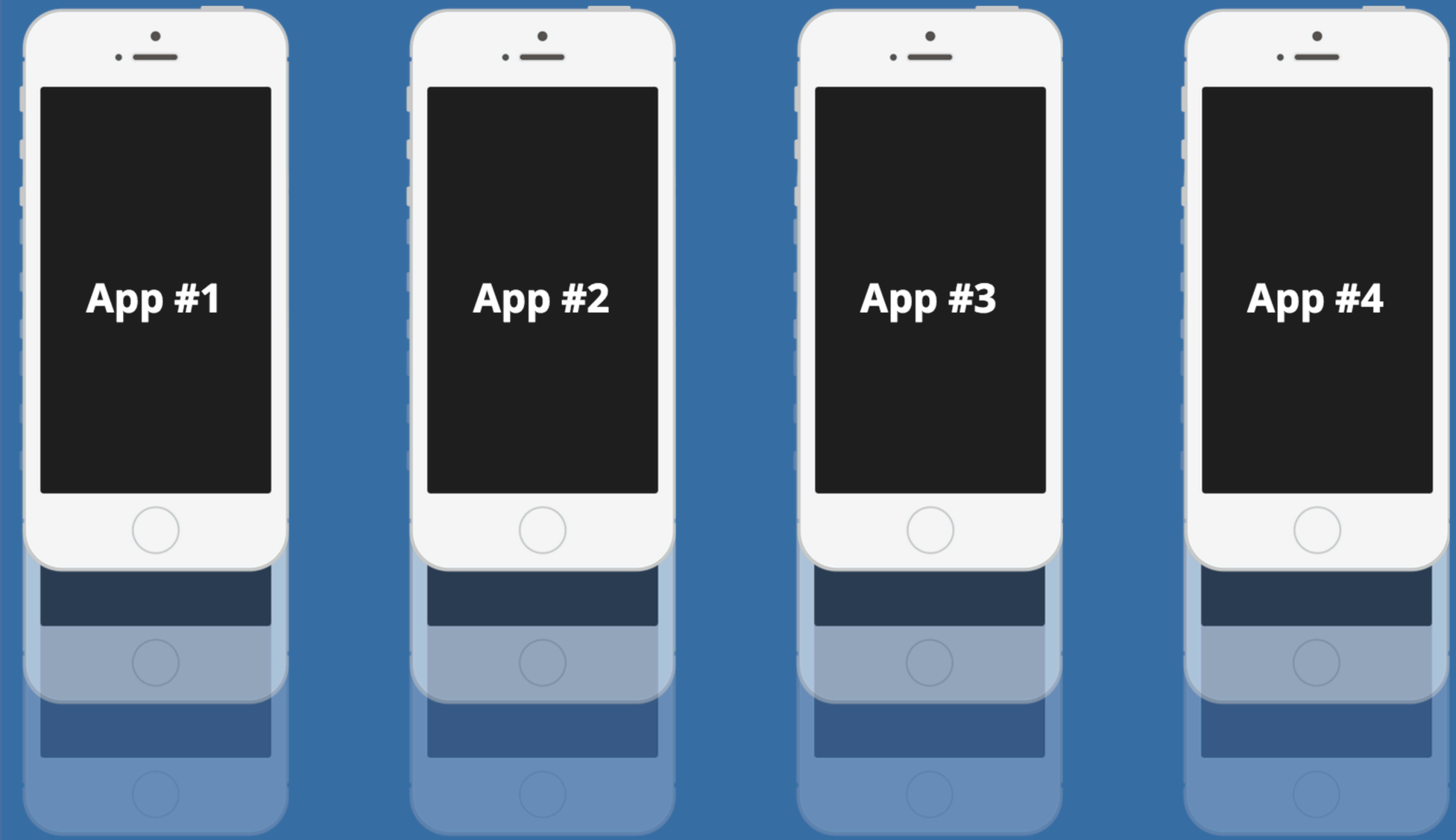

GET api/persons/1?include-films

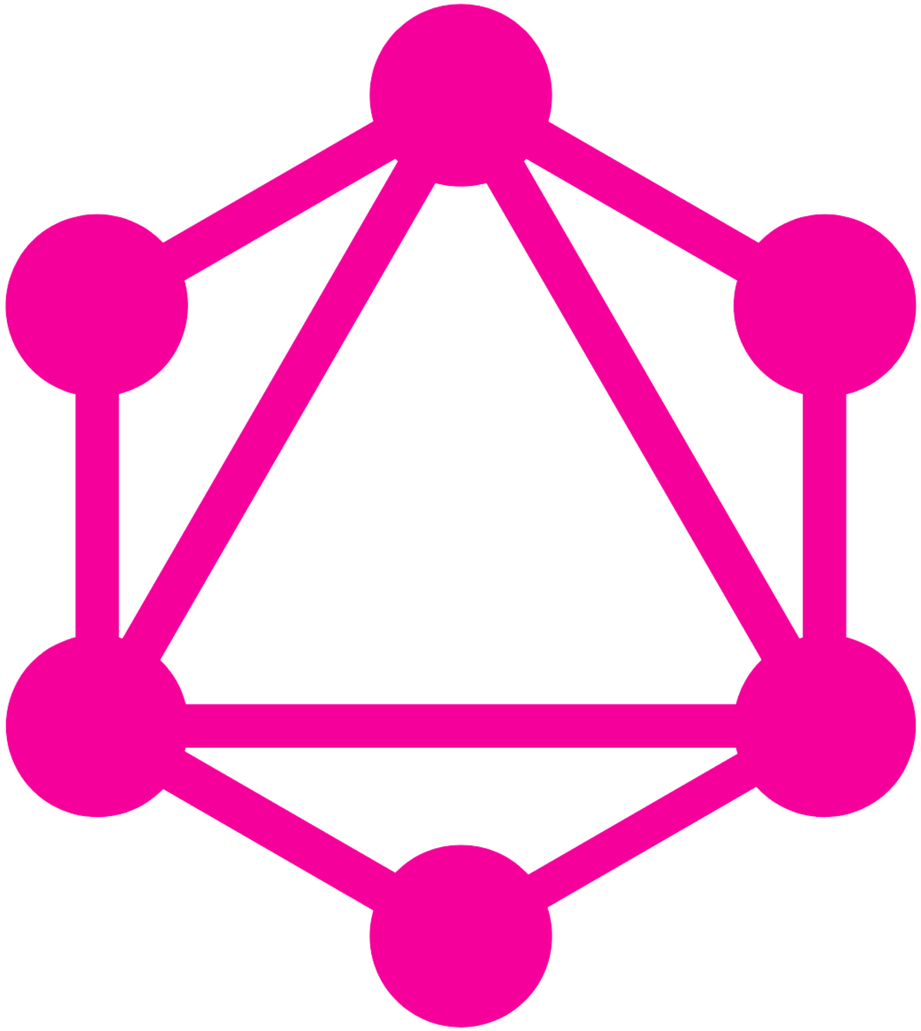
GET api/persons/1?include=films,...

Multiple apps



Multiple versions of multiple apps

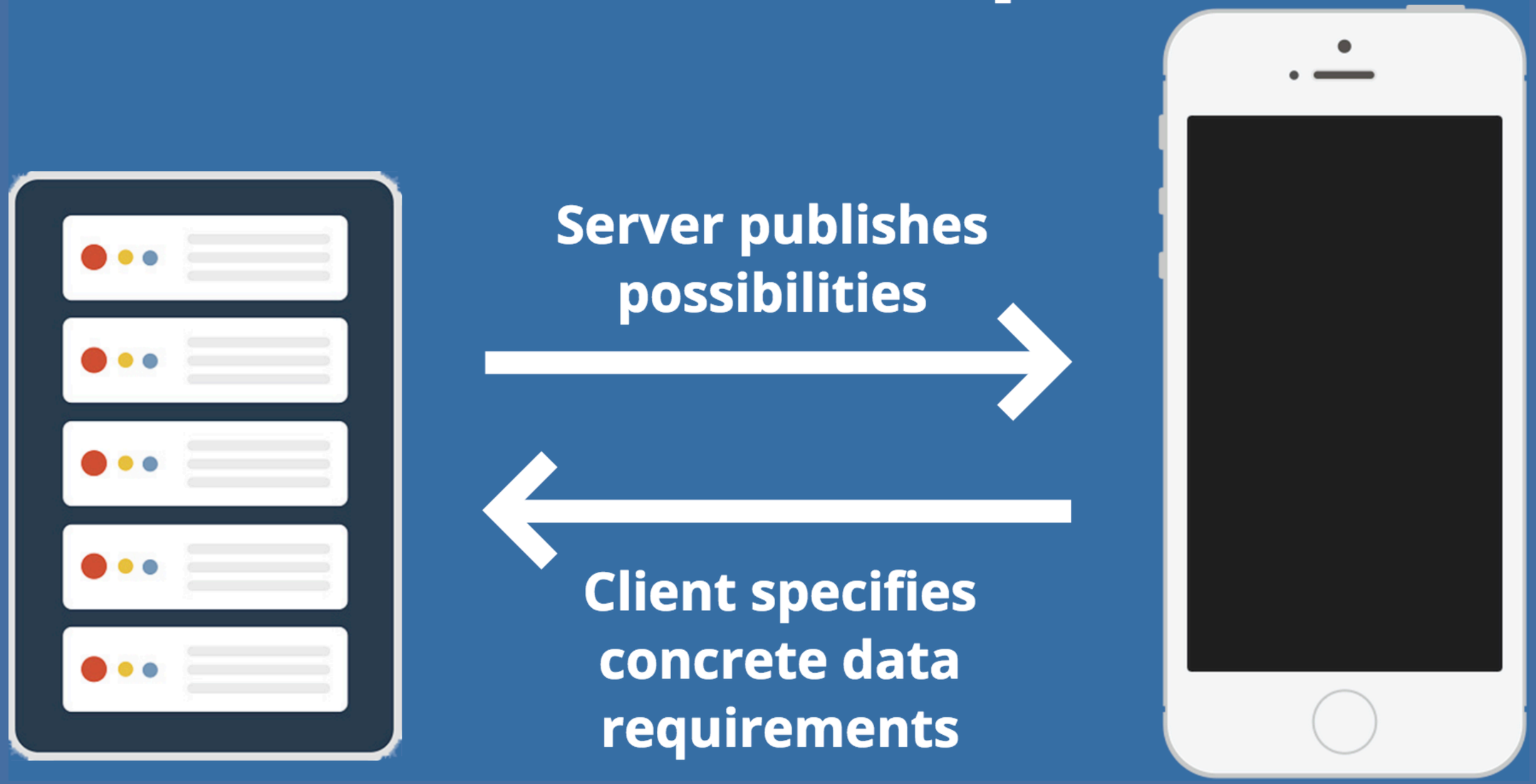




A data querying language!

Runs on arbitrary code …
Backed by a schema …
Based on a type system …
Making it fully introspective …
It is completely agnostic of your storage layer …

Evolving the server–client relationship



```
type Query {
  person(id: Int!): Person,
  ...
}

type Person {
  name: String!,
  gender: String,
  height: Int,
  films(first: Int, last: Int): [Film],
  ...
}

type Film {
  title: String!,
  persons(first: Int, last: Int): [Person],
  ...
}
```

Each schema is an arbitrarily nested hierarchy of type definitions.

Each GraphQL endpoint is using a well defined schema.

And the best part: each GraphQL endpoint by default exposes the Schema!

=> Introspection: Exposing your own schema through the type system.

```
{
  "data": {},
  "errors": []
}
```

```
{
  "data": {
    "createUser": null
  },
  "errors": [
    {
      "message": "Email already exists",
      "locations": [
        {
          "line": 32,
          "column": 11
        }
      ],
      "path": ["createUser.name"]
    }
  ]
}
```

GraphiQL

So what now?

The End